International Journal of Computer Networking,
Wireless and Mobile Communications (IJCNWMC)
ISSN(P): 2250-1568; ISSN(E): 2278-9448
Vol. 3, Issue 5, Dec 2013, 49-54

A REVIEW ON CONCURRENCY ISSUES IN MOBILE ENVIRONMENT

M. MOHANA[1] & C. JAYAKUMAR[2]

[1]Assistant Professor, Department of Information Technology, Easwari Engineering College, Tamil Nadu, India

[2]Professor, Department of Computer Science and Engineering, R.M.K Engineering College, Tamil Nadu, India

ABSTRACT

Due to the quick growth of wireless technologies and portable computers created a new working environment called mobile environment. This environment is different from the traditional distributed environment due to the following characteristics: the mobility of user or computers, frequent disconnections of wireless networks, and various energy constraints of mobile computing devices. These unique characteristics of mobile environment pose many challenges to the mobile transaction management, in terms of data inconsistency and data unavailability [8]. To preserve the consistency of data, most of the approaches use the traditional locking mechanism leads to blocking and increases the abort rate in mobile environment. Many researches has been focused on supporting these issues, still major issues have not been solved completely. This paper focuses on the complete survey of major issues concurrency in the mobile environment and the solutions that has been implemented to solve those problems.

KEYWORDS: Mobile Environment, Concurrency, Mobility, Mobile Transaction

INTRODUCTION

The mobile computing environment is emerged due to advancements in wireless networking technology and portable computing devices. Mobile computing enables users equipped with portable computing devices to access information services through a shared infrastructure, regardless of physical location or movement [6]. The mobile computing environment is a distributed computing platform with the following differences: the mobility of users and their access devices, frequent disconnection, limited bandwidth and the mobile resource constrains, limited computational and power sources. A mobile transaction processing system is a system that support mobile data sharing and to cope with the dynamic changes of mobile environments. At present, many types of mobile computing devices such as laptops and personal digital assistants (PDA) are available. The computing capacities of these mobile devices become more and more powerful in terms of processing speed, storage capacity and operating time. As a result, these mobile computing devices are becoming the essential work equipments. The rapid expansion of both the wireless network technologies and the capacity of mobile computing devices have created a new work environment. With the support of wireless networks and mobile computing devices, people can carry out their work while being on the move. The environment for accessing the information and processing is rapidly changing from stationary to mobile and location independent. This new work environment, called the mobile work environment, provides people a flexible and efficient work environment.

Transaction is a set of operations that translate a database from one consistent state to another consistent state. That is a computation processing is considered as a transaction or conventional transaction if it satisfies ACID (Atomicity, Consistency, Isolation, and Durability) properties. The simplest form of transaction is flat transaction. A mobile transaction is the transaction that can be executed and committed independent to its other parts. A transaction where at least one Mobile host

takes part in its execution is called mobile transaction, it very difficult to enforce ACID properties in mobile transactions due to the challenges such as the mobility of users and their access devices, frequent disconnection, limited bandwidth and the mobile resource constrains, limited computational and power sources, .Thus, new models are being created to deal with mobile transactions

Challenges of Transactions in Mobile Environments

Traditionally, database transactions with ACID (Atomicity, Consistency, Isolation, and Durability) properties have been used to enforce the integrity constraints of database systems in centralized or distributed environments. However, due to the challenging characteristics of the mobile environments such as the mobility of mobile computers, the frequent disconnections of wireless networks and the limited processing power of mobile computers the traditional database transactions may not be able to efficiently support transactions in volatile mobile environments. There are many new transaction models that have been developed to support transactions in mobile environments [9]. One common approach is to provide for the transaction processing systems adaptability to deal with different environment conditions and to cope with the constraints of mobile computing resources. However, there are still several major limitations. For example, the architecture of mobile transaction environments relies too much on the mobile support stations; a few research works focus on mobile transactions that are distributed among mobile hosts The ability to support both the disconnection and mobility is still a major challenge for mobile transaction models.

Mobile Database System

Mobile databases involve three parts namely fixed hosts, mobile units, and base stations. Fixed hosts perform the transaction and data management related functions with the help of database servers connected with the base stations. Mobile units are portable computers that move around a geographical region that includes the cellular network or “cell”. The cellular networks are used for communication between the mobile unit and the base stations. Base stations are two-way radios, which pass communications with the mobile units to and from the fixed hosts. When a mobile unit leaves a cell serviced by a particular base station that station transparently transfers the responsibility for the mobile unit's transaction and data support to whichever base station covers the mobile unit's new location.

TRADITIONAL DATABASE TRANSACTION CONCEPTS

A database is a collection of related data items accompanied by a data structure and a set of constraint rules that specify what information a data item represents. A database state is a collection of all the stored data values of all the data items in the database at a specific time. A consistent state of a database is a database state in which all the data values fulfill all the constraint rules of the database. A set of operations is usually provided to support users in retrieving or modifying data items in the database. These provided operations can be simple, for example read and write operations, or more complex operations, for example deletion or modification operations. To assist users to perform much more complex operations rather than reading from and writing to the database, a piece of specialized software called a database management system (DBMS) is accommodated to the database.

Database Transactions

Users can interact with the database by one or many database operations. The database operations can be gathered together to form a unit of execution program that is called a transaction. In other words, a transaction is a logical execution unit of database operations. A transaction transforms the database from one consistent state to another consistent state.

The ACID Properties

In a database system, there may be a large number of transactions that are executed concurrently, i.e., the shared data items in the database are read and possible written by many transactions at the same time. Each transaction must ensure that it always preserves the consistency of the database system. In order to retain and to protect the consistency of the database system, transactions will have the following ACID (Atomicity, Consistency, Isolation, and Durability) properties [9].

- **Atomicity:** Either all database operations of a transaction program are successfully and completely executed, or none of the database operation of this transaction program is executed.
- **Consistency:** A transaction must always preserve and protect the consistency of the database, i.e., it transforms the database from one consistent state to another. In other words, the result of a transaction that has committed fulfills the constraints of the database system.
- **Isolation:** An on-going transaction must not interfere with other concurrent transactions, or be able to view intermediate results of other concurrent transactions. In other words, a transaction is executed as if it is the only existing execution program on the database system at any given time.
- **Durability:** The result of a transaction that has successfully committed is permanent in the database. The consistent state of the database is always survived despite any type of failures

Concurrency Control of Transactions

Concurrency control is a database system techniques used to address the conflicts when more than one transaction is executed simultaneously. It controls the multiuser access of Database and preserves the correctness of data. Each transaction possesses the following characteristics:

- Transaction Ti starts by a Begin_transaction call is denoted by Bi.
- A database operation Opi(X) on a data item X is either a read operation Ri(X) or a write operation Wi(X). In general, more complex operations on a database system can be modeled via read and write operations.
- Transaction Ti ends by either a Commit transaction call denoted by Ci, or an Abort transaction call denoted by Ai.

The problems which are caused by the concurrent execution of transactions are: Lost Update, Dirty Read, and Unrepeatable Read. These problems are address as, the Lost update occurs when two transactions T1 and T2 try to write the same data item X. The transaction T2 overwrites the value of data item X that was prior written by transaction T1. The dirty read occurs when transaction T2 reads the value of data item X that is written by transaction T1 before the transaction T1 commits. If the transaction T1 aborts, the transaction T2 has been operating on an invalid data value. Finally, the unrepeatable read happens if a transaction executes the same read operation at different times, and obtains different data values.

The concurrency problems can be solved if the DBMS can schedule the operations of transactions in a way that no transaction interferes with other, i.e., fulfills the isolation property of transactions. The execution order that sequentially contains all the database operations of all concurrent transactions is called the schedule or history of transactions.

Concurrency Control Protocols

There are two main approaches for concurrency control protocols namely Pessimistic and Optimistic [1]. For the pessimistic approach, a database operation is checked if it could cause a non-serializable schedule before it is executed. The

database operation is rejected, i.e., the transaction is aborted, if it may potentially lead a schedule into a non-serializable schedule. For the optimistic approach, the submitted database operation is immediately executed as if there is no conflict between this database operation and database operations of other transactions.

Locking and timestamp ordering protocols are two common concurrency control protocols that are mostly used in the pessimistic approach. Concurrency control by the locking protocol requires that a transaction must request an appropriate lock on a data item before its database operation can be accepted for executing. In other words, a lock plays a role as an execution license for the database operation. One usually applies two types of lock: shared (read) and exclusive (write). A shared lock can be granted to many transactions at the same time, while an exclusive lock can only be assigned to one transaction at a time. Serializability among transactions can be guaranteed by a 2-phase locking (2PL) protocol. The 2PL protocol requires that a transaction must obtain all its locks (in growing phase) before it can release any lock (in shrinking phase). Strict 2PL is a locking protocol that only allows a transaction to release exclusive locks after it has committed or aborted.

Concurrency control by using timestamp ordering guarantees serializability among transactions based on the following time quantities: (1) the starting time or timestamp of each transaction TS, and (2) the read and write timestamp values for each data item X, denoted by Read_TS(X) and Write_TS(X) respectively. These read or write timestamp values correspond to the timestamp value of the latest transaction that successfully reads or writes the data item X. A timestamp can be a computer system clock or any logical counter maintained by the database system. When a transaction submits a database operation on a data item X, the timestamp TS of the transaction will be checked against the current read Read_TS(X) and write Write_TS(X) timestamp values of the data item. The outcome of this timestamp checking procedure is either the database system accepts the submitted database operation and the new timestamp value is updated for X, or the transaction is aborted.

MOBILE TRANSACTION AND CONCURRENCY MECHANISMS IN MOBILE ENVIRONMENT

Background

Transaction management on Mobile Database Systems (MDS) has to cope with a number of constraints such as limited bandwidth, low processing power, unreliable communication, and mobility etc. As a result of these constraints, traditional concurrency control mechanisms are unable to manage transactional activities to maintain availability. Innovative transaction execution schemes and concurrency control mechanisms are therefore required to exploit the full potential of Mobile Database System.

Concurrency Control is one of the basic building blocks of the transaction management. The characteristics of the mobile database environment make Concurrency Control a challenging issue. The mobile applications provides utmost convenience to the users, however they suffer from various limitations such as variable bandwidth, reduced storage, frequent disconnections, limited battery power etc. A crucial consideration in mobile databases is the ability to simultaneously access the data items irrespective of the physical locations of mobile users. To handle the concurrency control issue, various concurrency control techniques have been proposed in literature which are usually based on three mechanisms viz, locking, timestamps and optimistic concurrency control [2].

Concurrency control mobile transaction is used to increase throughput and allows timely and reliable access to shared data and must therefore support simultaneous execution and interleaving of multiple transactions. In mobile environment, the concurrency control algorithm has to overcome the effects of the local autonomy, in addition to the constraints imposed by the mobile units [8]. For an example, consider a transaction in a stationary computer on a wired network. The disconnection of network is often treated as a failure thus, when this occurs the execution of transaction is being

aborted. In a mobile environment, which is characterized by frequent disconnection (users may voluntarily disconnect, in order to conserve battery life), cannot be treated as a failure in the network. Transactions issued from mobile clients tend to be long-lived. Thus, transactions issued by mobile users are exposed to a larger number of disconnections. Another effect of long-lived transactions is that it could result in low system throughput. Long-lived transactions are more likely to result in conflicts. Pessimistic locking schemes in the implementation of concurrency control could result in blocking of concurrently executing transactions, resulting deadlocks and aborted transactions [8]. On the other hand, employment of optimistic concurrency control could result in a high rate of transaction restarts. Thus, still a new technique is needed in mobile environment that manages concurrency control and recovery, handles frequent disconnection, and to address the issue of long-lived transactions.

Related Works

In mobile computing environment, many mobile clients are concurrently accessing the database residing at the server, in the form of reads and writes. The broadcast-based data dissemination, in mobile computing systems, poses new challenging issues on data consistency of mobile transaction processing due to frequent disconnection from the network. Although data broadcast has been shown to be an efficient method for disseminating data items in mobile computing systems, the issue on how to ensure consistency and currency of data items provided to mobile transactions, which are generated by mobile clients, has not been examined adequately[6]. The authors discussed about the problem related to consistency and currency of data items provided to mobile transactions, which are generated by mobile clients and has not been examined effectively, the authors proposed model called smart server (SSM) and control the concurrency of mobile client’s reads and writes to provide consistent highly dynamic data to the mobile clients which are often disconnected from the network. The authors dynamic transmission disks (DTD) which are broadcasts to satisfy two types of users: frequently accessed data items user, rarely accessed data item users and also prioritize the reads and writes of mobile clients for maintaining consistency of the data provided to the mobile clients [6]. Smart Server Model with prioritized write back is proposed for mobile transaction is using Broadcast Marshaling in which data is accessed by their probability of it being accessed by the Clients but it is not discussed about what will be happening when two or more data have the same probability of it being accessed by the Clients. In this condition which data will be accessible first it needs to be discussed?

A Concurrency control in real time replicated databases (CCRTRD) mechanism uses S2PL (Static Two Phase Locking) for deadlock free environment [3]. The proposed mechanism has a considerably increased performance over S2PL which decreases execution time, waiting time of the current transaction and it also decreases the probability of deadlock. In CCRTRD protocol when a transaction requests a lock on an object held by other transactions in a conflicting lock mode, if the requester’s priority is higher than that of all the lock holders, the holders are restarted and the requester is granted the lock; if the requester’s priority is lower, it waits for the lock holders to release the lock. In addition, a new read lock requester can join a group of read lock holders only if its priority is higher than that of all waiting is written lock operations. In this protocol, if there are a large number of high priority request then the request which is being executed at first may have to wait for a long time [3]. Most of the metrics are not discussed in detail.

Lazy Database Replication with Snapshot Isolation [7], the author discussed about the problem related to snapshot isolation in concurrency controls and transaction inversions. In this paper, the author described architecture and algorithms which take advantage of the local concurrency controls to maintain global weak SI in lazy replicated systems. The author defined a new session-oriented correctness criterion called strong session SI and showed how it can be implemented efficiently to prevent transaction inversions in systems where each local concurrency control guarantees strong snapshot

isolation. The techniques use in this paper guarantee global weak SI and for avoiding transaction inversions through strong session SI in lazy replicated database systems. The weak SI system it assumes that read-only transactions are distinguished from update transactions in the request streams. Write transaction is not discussed in detail.

CONCLUSIONS

Even though many models have been proposed by many authors in mobile environment, the mobility brings in a novel dimension to the existing solutions to the problems in mobile database systems. We have surveyed a number of the problems and existing solutions. We have identified upcoming research areas that, due to the nature and constraints of mobile computing environment, need rethinking. The research directions discussed here will be the centre of attractions among the mobile database researchers in future.

REFERENCES

1. Jan Lindstrom, *"Optimistic Concurrency Control Methods for Real-Time Database Syatem",* Helsinki University Printing House,2003, ISBN 952-10-08776

2. EPFL, U. Grenoble, INRIA-Nancy, INT-Evry, U. Montpellier 2, U. Paris 6 and U. Versailles, *"Mobile Databases: a Report on Open Issues and Research Directions",* CNRS, the National Center for Scientific Research in France. December 2002.

3. Ashish Srivastava, Udai Shankar and Sanjay Kumar Tiwari, *"A Protocol for Concurrency Control in Real-Time Replicated Databases System",* IRACST – International Journal of Computer Networks and Wireless Communications (IJCNWC), ISSN: 2250-3501, Vol.2, No.3, June 2012.

4. KAM-YIU LAM, TEI-WEI KUO, WAI-HUNG TSANG, and GARY C.K LAW, *"Concurrency Control in Mobile Distributed Real-time Database Systems", Information Systems* Vol. 25 No. 4, pp. 261–286, 2000 ã 2000 Elsevier Sciences Ltd. All rights reserved Printed in Great Britain 0306-4379/00

5. Benimon N and Ashvanth B, *"Server Based MANETs with Updatable Cache for Consistency and Data Replication for Minimizing Network Contention",* International Journal of Computer Networking, Wireless and Mobile Communications (IJCNWMC), ISSN 2250-1568 Vol. 3, Issue 1, Mar 2013, 321-332

6. K. Chitra and Al-Dahoud Ali, *"Prioritized Transaction Management for Mobile Computing Systems"*, International Journal of Intelligent Computing Research (IJICR), Volume 1, Issue 4, December 2010.

7. Khuzaima Daudjee and Kenneth Salem, "*Lazy Database Replication with Snapshot Isolation*", VLDB '06, September 1215, 2006, Seoul, Korea. Copyright 2006 VLDB Endowment, ACM 1595933859/06/09.

8. K.Segun, A.R. Hurson, and A. Spink, "A Transaction Processing Model for the Mobile data Access System" LNCS 2127, pp. 112–127, Springer-Verlag Berlin Heidelberg 2001

9. Lim, J.B. ; Hurson, A.R. *"Data duplication and consistency in a mbile, multidatabase environment"* Proceedings. 1998 International Conference on Parallel and Distributed Systems